UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MEJORAMIENTO EN RECONOCIMIENTO DE VOZ MEDIANTE PREPROCESAMIENTO DE AUDIOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

Alejandro Mauricio Dumas Barrera

PROFESOR GUÍA:
Juan Barrios Martínez

MIEMBROS DE LA COMISIÓN
Mauricio Palma Lizana
Iván Sipiran Mendoza

SANTIAGO DE CHILE
2023

1. Resumen

Este informe presenta un trabajo de investigación e implementación de algoritmos que permiten mejorar el reconocimiento de voz en grabaciones de llamadas telefónicas.

La empresa Alloxentric tiene un producto que se encarga de analizar dichos audios, que son producidos en call centers de atención a clientes. Estas grabaciones son analizadas a través de sus transcripciones, y la transcripción de audios es una rama en la que se trabaja mucho actualmente y se encuentra en constante mejora. Muchas de las herramientas de transcripción actuales generan resultados con errores que no permiten un buen análisis de las grabaciones, lo cual se puede deber a poco entrenamiento de los modelos en conjuntos de audios específicos o a ruido en las señales de estos archivos. En base a este último escenario, es que surge la oportunidad de construir uno o varios algoritmos que permitan preprocesar los audios para así mejorar el reconocimiento de voz de las herramientas actuales de transcripción.

Con este objetivo, se implementaron dos algoritmos para cumplir con las siguientes tareas: diarizar las llamadas telefónicas para poder identificar en cada momento quien es el hablante en la conversación, y construir un modelo de inteligencia artificial que permita limpiar los ruidos más comunes en este tipo de audios.

Luego de realizar la diarización manual de un conjunto de 20 audios, se construyó un algoritmo de diarización que identificaba a los hablantes a través de los coeficientes RMS y MFCC de las señales de sonido, obteniendo unos prometedores resultados con un promedio de aciertos del 60 %.

Para la identificación de los ruidos, se analizaron 100 audios con malas transcripciones, y se encontró que los ruidos más comunes se debían a distorsiones aleatorias y eco dentro de las grabaciones. Con esto, se procedió a recrear dichos errores, y se entrenó una red neuronal WaveNET con las señales de audio sin procesar. Con esta herramienta, los audios fueron limpiados satisfactoriamente, pero los resultados al momento de comparar las transcripciones de los audios con y sin ruido fueron mixtos. Las diferencias entre los textos de cada una variaban poco en su mayoría, aunque hay algunas excepciones en donde la mejora es significativa, y motiva a seguir trabajando con el modelo implementado, ya sea modificando la estructura de la red neuronal, adaptando los hiperparámetros utilizados, o creando un dataset de entrenamiento con distorsiones más significativas que impacten la capacidad de los STT actuales.

Tabla de contenido

Capítulo I

2. Introducción

2.1. Contexto

La transcripción de audio es la conversión del contenido del discurso en un archivo de sonido a un texto escrito (Speech-to-text por sus términos en inglés), y es una práctica muy útil para mantener registro de ciertos eventos, almacenar información en un formato más simple, poder analizar discursos o conversaciones, entre muchas otras aplicaciones. Este proceso es muy costoso para ser realizado por seres humanos, y debido a esto, durante las últimas décadas ha sido de gran interés para la industria del desarrollo de software, ya que tecnologías que involucran inteligencia computacional han permitido avances muy importantes con resultados cada vez más precisos.

Alloxentric es una empresa startup dedicada a la generación y análisis inteligente de las comunicaciones, integrando voz y texto para diseñar y evaluar las mejores estrategias de contactabilidad. La plataforma ofrece servicios que involucran ponerse en contacto con personas, tanto por llamadas como por mensajes (por ejemplo, bots de voz/chat), y también ofrece servicios donde se deben analizar conversaciones de forma automática (por ejemplo, evaluar servicios de call-centers con clientes). Para esto, Alloxentric necesita una robusta infraestructura de software que les permita tanto convertir archivos de audio a texto, como también crear audios a partir de texto.

2.2. Problema y relevancia

En este contexto, Alloxentric ha encontrado un cuello de botella al momento de generar las transcripciones de conversaciones, ya que a medida que va obteniendo un volumen más grande de audios, se encuentran cada vez más errores y diferencias entre lo que se puede escuchar y lo que se obtiene como texto. Considerando que el flujo de audios que maneja la empresa es del orden de cientos de nuevos audios (por cliente) cada día, estos errores en las transcripciones provocan grandes limitaciones al momento de querer analizar dichos audios en su servicio de evaluación. En particular, se requiere analizar las conversaciones que sostienen agentes de call-centers con sus usuarios, y así poder evaluar el desempeño de los agentes, y la calidad del servicio prestado. Esto se realiza buscando palabras clave dentro de lo que dijo el agente para identificar si cumplió con el guión que debe seguir o no. Los errores en las transcripciones dificultan esta evaluación, debido a que las palabras clave deben ser escogidas considerando dichos errores, haciendo la tarea más compleja de lo que ya es.

Actualmente, las transcripciones son imprecisas, y esto se puede atribuir a diferentes causas:

- Imprecisiones del algoritmo entrenado para realizar la transcripción.
- Múltiples personas hablando al mismo tiempo.
- Mala calidad del audio (bajo volumen, ruido de fondo, etc.).

Por esto, el trabajo de título propuesto en este documento tiene como finalidad poder

disminuir las imprecisiones encontradas en las transcripciones en el contexto de llamadas telefónicas sostenidas por agentes de call-centers. Para lograr esto, se realizará un trabajo de investigación para conocer las diferentes herramientas de reconocimiento de voz (ASR por las siglas en ingles de Audio Speech Recognition), evaluando su uso y desempeño en la conversión a texto de los audios de interés de la empresa Alloxentric, y definiendo las distintas adaptaciones que deberán ser realizadas para mejorar dicho performans.

2.3. Objetivos

Objetivo General El objetivo principal del proyecto es lograr un mejoramiento de la calidad de reconocimiento de voz mediante el preprocesamiento de los audios e identificación de las personas participantes, en el contexto de grabaciones de llamadas telefónicas de call-centers. Con esto se busca realizar transcripciones (con Deepgram principalmente) de mejor calidad, obteniendo conversiones a texto más precisas.

Objetivos Específicos

1. Crear un algoritmo de reconocimiento de voz que permita identificar a las personas participantes dentro de una conversación, separando lo que dice cada uno.
2. Identificar las fuentes más comunes de ruido en los audios, y entrenar un modelo de machine learning que mejore su calidad reduciendo dicho ruido.
3. Realizar una comparación en la calidad de las transcripciones entre audios originales y audios con ruido reducido y diarización.

2.4. Metodología

El trabajo realizado en esta memoria comenzó con un estudio bibliográfico de cual es el estado del arte de los algoritmos y procedimientos comunmente utilizados en las transcripciones de archivos de audios, junto a un analisis de las principales herramientas para procesar este tipo de ficheros.

Luego de esto, la segunda etapa consistió en construir el dataset a utilizar, para lo cual fue necesario definir y escoger el conjunto de audios y realizar la transcripción manual de un determinado grupo.

En la tercera etapa se implementaran dos algoritmos, uno para diarizacion y otro para la eliminacion de ruido. El segundo sera entrenado y finalmente ambos seran analizados para comprobar su eficacia en sus respectivas tareas.

2.5. Descripción general de la solución

Se comenzó construyendo un algoritmo sencillo de diarizacion utilizando las librerias de procesamiento espectral de audios disponibles en Python. Para esto, fue necesario realizar la transcripción manual de 20 audios correspondientes a conversaciones telefonicas y a traves del analisis de diversos coeficientes obtenidos de su analisis espectral, clasificar cual es el hablante más probable en cada segundo de la conversacion.

Despues de esto, se comenzó a implementar un algoritmo de redes neuronales que permitiera hacer una limpieza del ruido dentro de un archivo de audio. Para cumplir con esto, primero fueron definidas cuales señales son interpretadas como ruido, se seleccionó un dataset de 1000 audios con el cual fue entrenado el modelo, y finalmente se hicieron validaciones con un conjunto más pequeño de 50 audios que debieron ser transcritos manualmente.

Con estos, el modelo de limpieza de ruido fue evaluado transcribiendo con la plataforma Deepgram los audios sin limpiar y comparándolos con la transcripción de la misma herramienta pero con el audio procesado.

Capítulo II

3. Estado del Arte

3.1. Espectro de ondas

El espectro de una onda de sonido corresponde a una representación de dicha onda que nos permita entender y diferenciarla de otras, pudiendo encontrar patrones o identificadores importantes en su clasificación y posterior comparacion.

Cuando hablamos del espectro de una onda, podemos encontrar dos definiciones importantes:

- Forma de la onda: Esta corresponde al grafico de la amplitud de la onda en funcion del tiempo, mostrandonos como varia la intensidad del sonido, y dandonos una imagen entendible para nosotros de como representar cierto archivo de sonido.
- Espectro de frecuencia: Esta corresponde a la cantidad de energía que hay en cada componente de frecuencia de la señal. En un gráfico típico de espectro, la frecuencia se coloca en el eje horizontal (generalmente en hercios, Hz) y la amplitud (o intensidad) se coloca en el eje vertical. Esto nos permite ver claramente las frecuencias dominantes y la intensidad con la que contribuyen a la señal de sonido. Son imagenes que podemos determinar a cada momento, y nos entregan más información de como esta constituida la onda.

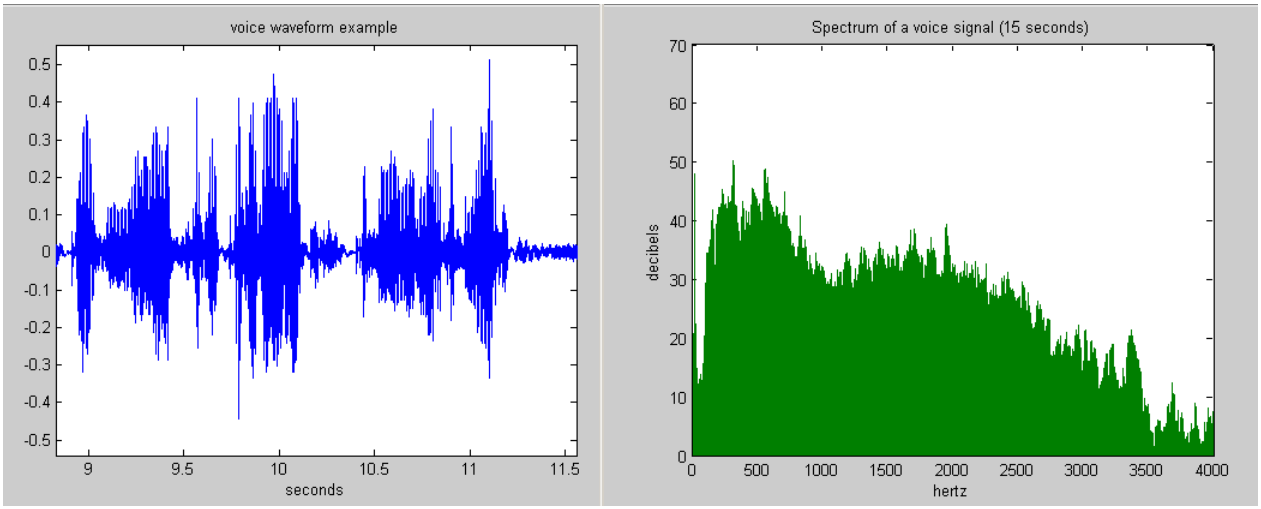

Figura 1: Izquierda: Imagen de la forma de una onda. Derecha: El espectro de frecuencia de un determinado momento.

3.1.1. Transformada de Fourier

La Transformada de Fourier es una herramienta matemática fundamental en el análisis de señales y el procesamiento de señales, y se utiliza para descomponer una señal en sus componentes de frecuencia. Permite analizar una señal en el dominio de la frecuencia, lo que significa que puedes ver qué frecuencias están presentes en la señal y cuánta energía hay

en cada una de esas frecuencias. Es la herramienta utilziada para calcular el espectro de frecuencias mencionado anteriormente.

La Transformada de Fourier toma una señal en el dominio del tiempo (una función del tiempo) y la convierte en el dominio de la frecuencia. En otras palabras, descompone una señal en una suma de senoides (funciones seno y coseno) a diferentes frecuencias.

3.1.2. Coeficientes y características

En el análisis espectral de una onda de sonido, hay varios coeficientes y características importantes que se pueden extraer para describir las propiedades de la señal de audio. Estos coeficientes y características proporcionan información sobre la energía, el tono, la forma y otras características relevantes de la señal. Para el caso de este trabajo, nos enfocaremos en dos que serán los más relevantes:

RMS: El coeficiente RMS (Root Mean Square, en inglés), en el contexto del análisis espectral de una señal, es una medida estadística que se utiliza para cuantificar la amplitud (o intensidad) de una señal de forma más precisa que simplemente calcular el valor promedio. Se aplica comúnmente en el análisis de señales, como señales de audio o señales eléctricas, para caracterizar su nivel de potencia o intensidad.

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\left({x_1}^2 + {x_2}^2 + \cdots + {x_n}^2\right)}.$$

Figura 2: Fórmula matemática para calcular el RMS, en donde x_i corresponde al valor de la amplitud de la onda en el tiempo i

El coeficiente RMS representa la raíz cuadrada de la media de los valores cuadrados". Al elevar al cuadrado los valores antes de tomar la media, esto resalta los valores más grandes y penaliza los valores más pequeños, lo que resulta en una medida que representa la potencia o la intensidad efectiva de la señal. Esto es especialmente útil cuando se trabaja con señales que varían en amplitud con el tiempo, como las señales de audio.

MFCC: Los coeficientes MFCC (Mel Frequency Cepstral Coefficients, por sus siglas en inglés) son características ampliamente utilizadas en el análisis espectral de señales de audio, especialmente en tareas de procesamiento de voz y reconocimiento de voz. Estos coeficientes representan de manera eficiente las características relevantes del espectro de frecuencia de una señal de audio y se derivan a partir de la escala de frecuencia mel, que es una escala perceptual de frecuencias sonoras.

Los coeficientes MFCC se utilizan en una variedad de aplicaciones de procesamiento de voz y análisis de audio, incluyendo:

- Reconocimiento de voz: Los coeficientes MFCC son ampliamente utilizados en sistemas de reconocimiento de voz automáticos para identificar palabras y frases en señales de audio.

- Clasificación de hablantes: También se utilizan para identificar las características únicas de diferentes hablantes en el análisis de voz.
- Filtrado de ruido: Pueden ser utilizados para detectar y filtrar ruido no deseado en señales de audio.
- Síntesis de voz: En la síntesis de voz, los coeficientes MFCC se utilizan para generar voces sintéticas con características naturales.
- Música y análisis de sonido: También se utilizan en aplicaciones de análisis de música y sonido, como la clasificación de género musical, el reconocimiento de instrumentos, y más.

3.2. Reconocimiento del habla (ASR)

El Reconocimiento Automático del Habla (RAH), también conocido como ASR (Automatic Speech Recognition, en inglés), es una tecnología que permite a las computadoras o sistemas informáticos convertir el habla humana en texto escrito de forma automatizada y precisa. Es decir, el RAH permite que las máquinas comprendan y transcriban lo que una persona dice en lenguaje natural.

Es una disciplina de la inteligencia artificial que ha avanzado significativamente en las últimas décadas gracias a los avances en técnicas de procesamiento de señales, aprendizaje automático y redes neuronales profundas, lo que ha mejorado la precisión y la aplicabilidad de esta tecnología en diversas áreas.

3.3. Transcripciones

En la actualidad existen múltiples servicios que permiten realizar transcripciones de audios, y entre ellos se pueden encontrar tanto plataformas gratuitas (muchas veces como demostraciones limitadas) como aplicaciones de pago. Los idiomas en los cuales se han desarrollado estas herramientas también son variados, pudiendo realizar transcripciones en español, inglés, portugués, entre otros (siendo estos los más relevantes para nuestro sector del mundo). Las principales limitaciones e imprecisiones que tienen las herramientas actuales de transcripciones son que muchas no se encuentran optimizadas para entender nombres propios o expresiones en español, además de ser muy sensibles ante el ruido o la baja calidad de los audios, y ambas cosas son muy comunes en las grabaciones de llamadas telefónicas realizadas desde call-centers, las cuales son el foco de esta investigación. Actualmente, dentro de la empresa Alloxetric se utiliza una aplicación externa que entrega el servicio de transcripción de audios, la cual se llama Deepgram [2]. Esta aplicación permite transcribir archivos de audio en español, aunque no está completamente optimizado para esta lengua, ya que su principal foco es el idioma inglés. Los errores son más marcados cuando los participantes de la comunicación hablan muy rápido, cuando se nombran sustantivos propios (como nombres) o cuando se habla sobre números.

Un problema común que surge es la identficación de cada participante de la conversación, y para la empresa es algo deseable debido a que busca evaluar solo el desempeño de uno de los participantes. En esta área, no existen herramientas que cumplan de manera satisfactoria con esta misión, y si bien la aplicación Deepgram tiene la posibilidad de determinarlos, no logra realizarlo de buena manera, ya que es común que corte las oraciones de los participantes antes

de que estos terminen, y se las asigne incorrectamente al otro. Esto es probablemente debido a la naturaleza de las grabaciones, las cuales pueden dficultar la tarea si no se desarrolla un software específico.

Otro problema que se encuentra al analizar este tipo de audios es el nivel de ruido. Entre las principales causas pueden estar:

- Bajo volumen de las grabaciones.
- Sonidos de fondo como vehículos, artículos electrónicos, herramientas encendidas, otras personas conversando, etc.
- Múltiples integrantes de la conversación hablando a la vez.

3.4. Diarización

La diarización corresponde a la identificación de los hablantes en una conversación, indicando que participante está hablando en cada segundo de una grabación de audio. El objetivo es obtener al final una separación de lo que dijo cada hablante, junto con el orden en el cual fueron interviniendo.

Para lograr esto, los métodos comunmente constan de los siguientes pasos:

- Detección de voz, en donde se necesita algún módulo que permita separar la conversación de lo que no es conversación, para obtener un audio sin silencios ni sonidos que no pertenezcan a los hablantes.
- Segmentación de voz, en donde se divide el audio en segmentos pequeños (comunmente de 1 segundo) con el objetivo de que cada segmento de voz pertenezca solo a uno de los hablantes.
- Generar vectores que representen esos segmentos de voz, que puede ser realizado a través de cálculos sobre la FFT de los audios o basados en las transformaciones de una red neuronal específica.
- Clustering, que sería poder agrupar los vectores que definen a los segmentos por cercania, y asignándoles una etiqueta del hablante. Con esto, obtenemos a quien pertenece cada uno de los segmentos del audio.

Un trabajo de investigación relevante en este apartado fue publicado por Google Brain y se llama *Speaker Diarization with LSTM* [8]. Como su nombre lo indica, utiliza una red neuronal LSTM (por las siglas en inglés de Long Short Term Memory) para generar los vectores que representan a cada uno de los segmentos. Las redes LSTM poseen bucles de información que permiten recordar estados previos y asi decidir los siguientes estados de la iteración (lo que sería como una memoria de más largo plazo).

En el desarrollo de este paper, se utilizó parte de la librería de **pyannote** para separar la parte del archivo de audio que contenía la conversación de la que no. Luego se realizó la segmentación del audio en ventanas que tenian solapamiento (por ejemplo, Ventana1=[0, 1], Ventana2=[0.5, 1.5], Ventana3=[1, 2], etc.). La definición de los vectores se realizó a través del cálculo de coeficientes FFT que posteriormente recorrerieron la red LSTM ya mencionada. Y finalmentente, se compararon distintos métodos de clustering, pudiendo destacar al algoritmo

de **K-means** como el con mejores resultados.

El método descrito anteriormente fue probado con 3 datasets públicos de audios en inglés, y en este lenguaje probó realizar diarizaciones con muy buenos resultados. Otro trabajo que es relevante en esta área de investigación corresponde a **pyannote-audio** [1], el cual fue nombrado en el paper anterior, y corresponde a una librería de código abierto que ayuda en el pre-procesamiento de audios, y también implementó un algoritmo de diarización. Este es especialmente interesante porque dentro de su set de herramientas, provee de modelos pre-entrenados y de métodos de re-entrenamiento para adecuar el modelo a diferentes casos de uso.

Esta librería está escrita en python, y utiliza el framework de machine learning **pytorch** para los modelos de redes neuronales que utiliza. Dentro de sus herramientas provee de modelos para reconocer voces, reconocer conversaciones, detectar cambios de hablantes y voces solapadas, además de ayudar con el procesamiento de audios como la obtención de vectores que representen segmentos de señales.

Su única desventaja es que sus modelos han sido entrenados con datasets más pequeños, y todos con ejemplos en inglés, pero permite el re-entrenamiento y especialización de su algortimo.

3.5. Que consideramos ruido

En el contexto del procesamiento de señales de audio, el término **ruido** se refiere a cualquier sonido no deseado o no relevante que se mezcla con la señal de audio principal. El ruido puede tener diversas fuentes y características, y puede afectar negativamente la calidad y la inteligibilidad de la señal de audio.

Existen diferentes tipos de ruido en el ámbito del audio, algunos de los más comunes son:

- Ruido ambiental: Es el ruido de fondo presente en el entorno de grabación, como el ruido de tráfico, murmullos, acondicionadores de aire, ventiladores, etc. Este tipo de ruido puede afectar la calidad del audio y dificultar la comprensión de la señal principal.
- Ruido de fondo electrónico: Se debe a interferencias electromagnéticas o problemas en los dispositivos de grabación o reproducción, como zumbidos, estática o ruidos de alta frecuencia.
- Ruido de cuantización: Es un tipo de ruido introducido durante la digitalización de la señal analógica de audio. Ocurre cuando la resolución de bits utilizada para muestrear la señal es insuficiente, lo que puede resultar en distorsiones y artefactos.
- Ruido de banda ancha: Es un ruido que cubre un amplio rango de frecuencias y puede ser causado por diversos factores, como la mala calidad del equipo de grabación o la amplificación de la señal.

La reducción de ruido es un proceso que implica identificar y separar el ruido de la señal principal y luego atenuar o eliminar el ruido mientras se mantiene la integridad de la señal deseada. Este proceso puede ser útil en diversas aplicaciones, como la mejora de grabaciones de audio, la eliminación de ruido de fondo en llamadas telefónicas, entre otros casos.

3.6. Reducción de ruido

Investigando sobre las soluciones que existen en el area de disminución de ruido en archivos de audio, se pueden encontrar de 2 tipos principalmente:

- Inteligencia artificial para la detección de ruido.
- Métodos de "spectral gating"para quitar un determinado espectro de sonido de la grabación.

Dentro del primer tipo de soluciones se encuentra el trabajo **DeepFilterNet** realizado por Hendrik Schroter [7], el cual plantea una red neuronal con un diseño de baja complejidad, con el cual se puede realizar el mejoramiento de las grabaciones con una técnica computacional más eficiente que otras soluciones. Este algoritmo está escrito en python, utilizando las herramientas de pytorch y pytorchaudio, y su autor lo publicó para ser utilizado como una librería desde python. Posee un modelo pre entrenado con datos de grabaciones en inglés, y permite el re-entrenamiento de su modelo con otros datos en un formato especifico.

Para el segundo tipo de soluciones, podemos destacar el trabajo de Tim Sainburg con el algoritmo de **NoiceReduce** [5], desarrollado para mejorar los audios de unas grabaciones de animales para el artículo de biologia *Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires* [6]. En este caso, lo que se busca es construir un archivo de audio que solo contenga el ruido que se desea remover, para luego construir un filtro que podamos aplicar a la señal que deseamos limpiar.

Los pasos de este algoritmo son:

- Calcular la transformada rápida de Fourier (FFT) sobre el ruido.
- Calcular estadísticas sobre la FFT del ruido y obtener un set de parámetros.
- Se calcula un límite sobre este set de parámetros para indicar cuando nos encontramos con una señal de ruido y cuando no.
- Calcular la FFT del archivo de audio que deseamos limpiar.
- Determinar una máscara o filtro de señal comparando la FFT del audio con el límite.
- Aplicar la máscara de señal al FFT del audio.
- Invertir la FFT del audio para volver a obtener un archivo de sonido.

3.7. WaveNET

WaveNET es una arquitectura de redes neuronales profundas especializada en la generación de **raw audio waveforms** (ondas de audio sin procesar). Fue diseñada por un grupo de investigadores de Google DeepMind, publicado en el paper *WAVENET: A GENERATIVE MODEL FOR RAW AUDIO* [3] en el año 2016. La diferencia de este tipo de red neuronal con otros modelos utilizados en el procesamiento de audios es que puede trabajar con la onda de sonido sin procesar, evitando la definición de un vector de características que defina a la onda en cada segundo. Esto tiene la ventaja de generar señales de audio con menos errores al evitar la recostrucción del sonido desde el vecrot de características, y permite un entrenamiento más sencillo, con menos variables que definan la señal de audio a cada segundo.

Entrando en su arquitectura, las redes neuronales WaveNET se componenen principalmente de capas convolucionales causales, las cuales son capas convolucionales en donde no se puede alterar el orden de la información. Las predicciones de cada paso de tiempo en el modelo solo dependen de los pasos anteriores, y no de la forma de la onda en los pasos futuros. Esto permite un entrenamiento más rápido, especialmente cuando se trabaja con inputs de secuencias muy largas.

Además de lo anterior, con el objetivo de poder aumentar el campo receptivo de la red neuronal, es decir, poder agregar a la predicción de cada punto la mayor información del resto de la onda posible, se utiliza una convolución retrasada (Dilated Causal Convolutional). Esta es una capa en donde la convolución se realiza entre puntos alejados entre si (definidos por el largo del retraso), saltándose valores del input y permitiendo que el resultado de cada punto no dependa solo de sus vecinos más cercanos, sino que se vea afectado por puntos más distantes. Si bien este aumento del campo receptivo puede ser logrado agregando más capas convolucionales al modelo, las capas convolucionales dilatadas ofrecen una forma más eficiente de hacerlo, disminuyendo los tiempos de entrenamiento y predicción de la red neuronal.

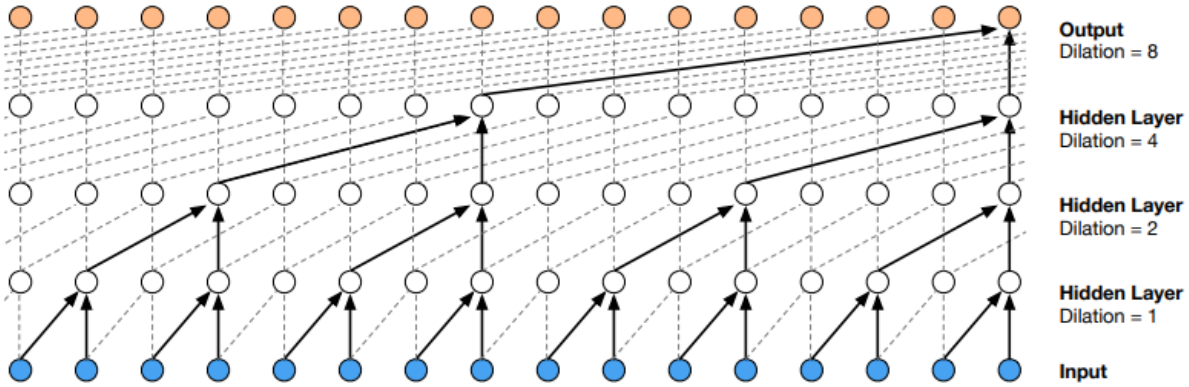

Figura 3: Visualización de un conjunto de capas Dilated Causal Convolutional [3]

Debido a que el objetivo de la red neuronal WaveNET es la generación de una señal de audio con la misma dimensionalidad temporal que la señal de input, no se usan capas de Pooling dentro del modelo, que son un tipo de capa utilizada para reducir la dimensionalidad de un arepresentación generada por una capa convolucional.

Capítulo III

4. Solución

El desarrollo de este trabajo tiene la finalidad de mejorar la calidad de las transcripciones de audios correspondientes a grabaciones de llamadas telefónicas, y para esto se dividirá la investigación en dos secciones:

- Diarización: Implementar un algoritmo de reconocimiento de voz que permita identificar a los participantes dentro de una conversación, separando lo que dice cada uno.
- Limpieza de ruido: Identificar las fuentes más comunes de ruido en los audios, y entrenar un modelo de machine learning que mejore su calidad reduciendo dicho ruido.

4.1. Diarización

Se definió un set de 20 audios con grabaciones de llamadas telefónicas de call-centers, y se visualizaron diferentes características:

1. Separación entre silencio y conversación.
2. Cálculo del espectro de señal del audio.
3. Cálculo de coeficientes MFCC (Mel Frequency Cepstral Coeficient).

Se hizo una diarización manual de 20 audios para comenzar a buscar diferencias entre los parámetros calculados, las cuales se almacenan en el siguiente formato:


```
{
  "AudioNombre": "dserna_29-08-2022_132020_0972022313.mp3",
  "TiempoAudio": 85.175934,
  "TranscritoChat": [
    {
      "speaker": 0,
      "start": 2.0971153,
      "end": 2.7362363,
      "transcript": "Buenas tardes."
    },
    {
      "speaker": 1,
      "start": 3.4951923,
      "end": 4.174258,
      "transcript": "Buenas tardes."
    },
    {
      "speaker": 0,
      "start": 4.9731593,
      "end": 7.585,
      "transcript": "Doña SASKIA NATALI Palacios Pérez?"
    },
    {
      "speaker": 1,
      "start": 8.385,
      "end": 8.625,
      "transcript": "No."
    }
  ]
}
```


Figura 4: Formato de Diarización de un audio transcrito

Aqui podemos ver que se almacena el nombre del audio, el tiempo de duración en segundos, y una lista con la conversación separando las frases que dijo cada hablante (o speaker) y el tiempo de cuando comenzo a hablar y cuando termina.

El análisis de estos audios se realizará en el campo de frecuencia, por lo que es indicado poder graficar esta característica en función del tiempo.

Junto con esto, para poder calcular los coeficientes que sean representativos del audio, es necesario segmentar el archivo para poder determinar sus parámetros en cada intervalo.

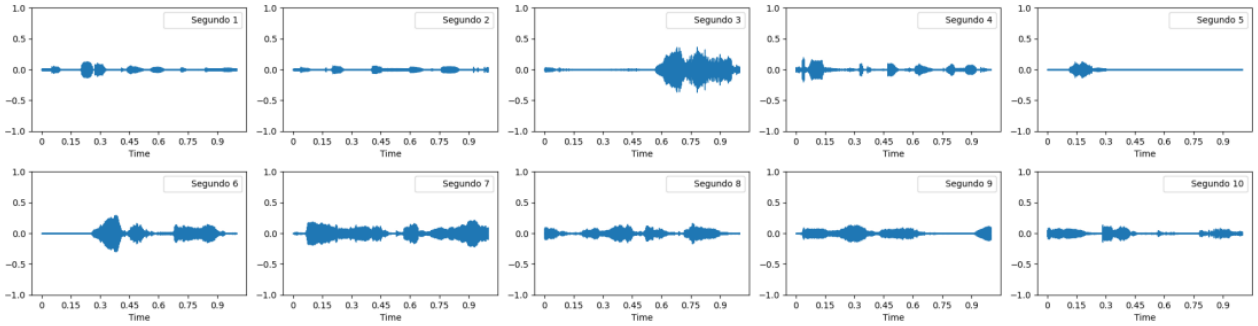

Figura 5: Espectro segmentado de un audio

Para cada uno de los audios tenemos una diarización, con la cual podemos graficar como es que va variando el hablante en función del tiempo. Con esto, podremos comparar si alguna de las variaciones de los parámetros que identifican un intervalo de audio se correlaciona con los cambios entre hablantes.

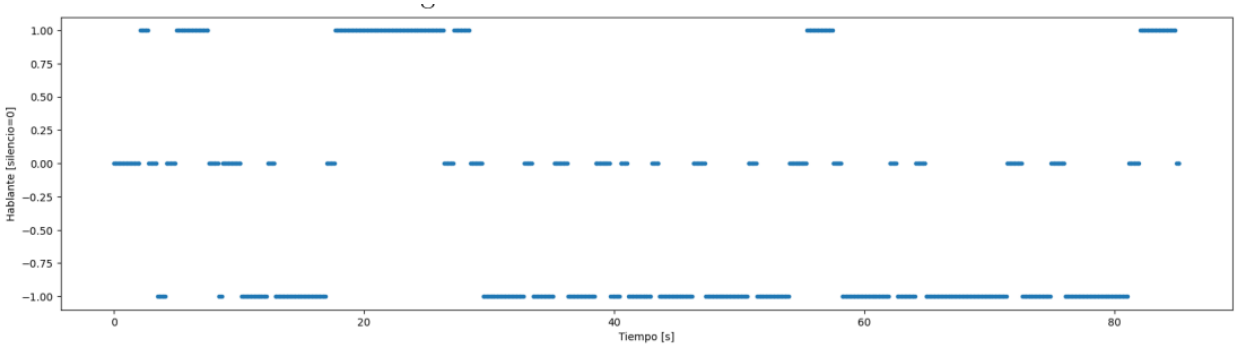

Figura 6: Diarización de un audio

Para el trabajo se calcularon un gran número de coeficientes en cada uno de los audios, los cuales son:

- Zero Crossing Rate
- Spectral Centroid
- MFCC
- Chroma Cens, Sqt y Stft
- Mel Spectrogram

- Poly Features
- RMS

Y con cada uno de esos parámetros calculados, se buscó establecer alguna relación con el hablante de la diarización.

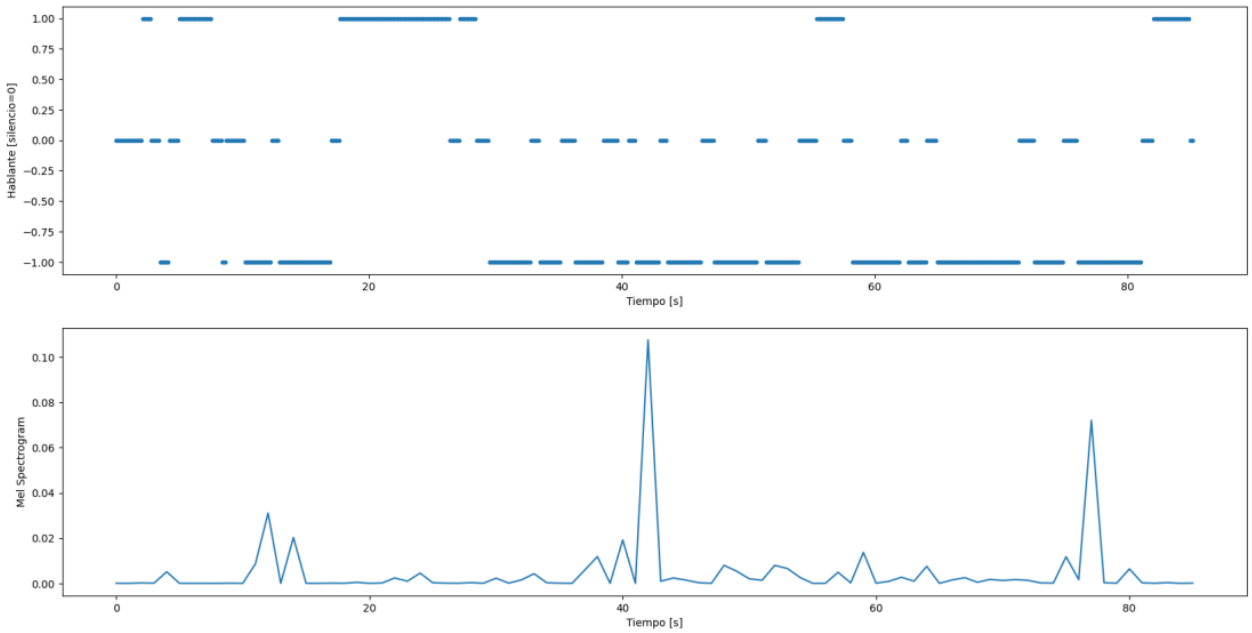

Figura 7: Diarización y Mel Spectrogram

Después de investigar y experimentar con los coeficientes calculados, se determinó que los parámetros que más variaban al cambiar el hablante de la conversación correspondian al RMS y al MFCC. El RMS está relacionado con la potencia de la señal del audio, y el MFCC es un grupo de coeficientes que están ligados al tono de la voz.

Luego de eliminar los intervalos de silencio de la conversación, se procedió a graficar como es que estos dos coeficientes varian en función del tiempo, obteniendo lo siguiente:

En el gráfico se aprecia una diferencia de segundos en el eje x de la diarización, la cual es producto de los cortes abruptos que realiza la librería pydub al separar el silencio de la conversación.

Utilizando la información de ambos coeficientes, se realizó una diarización a partir del RMS y del MFCC de la conversación. Para esto, se siguieron los pasos:

1. Calcular el promedio del parámetro en el tiempo.
2. Separar los puntos identificando cuales están sobre y bajo el promedio.
3. Suavizar el ruido agrupando los puntos según su vecindad más próxima.
4. Resolver la diferencia en tiempo entre los puntos de cada parámetro y la diarización manual, agregando puntos al inicio y final de cada una de las frases identificadas para cada hablante.

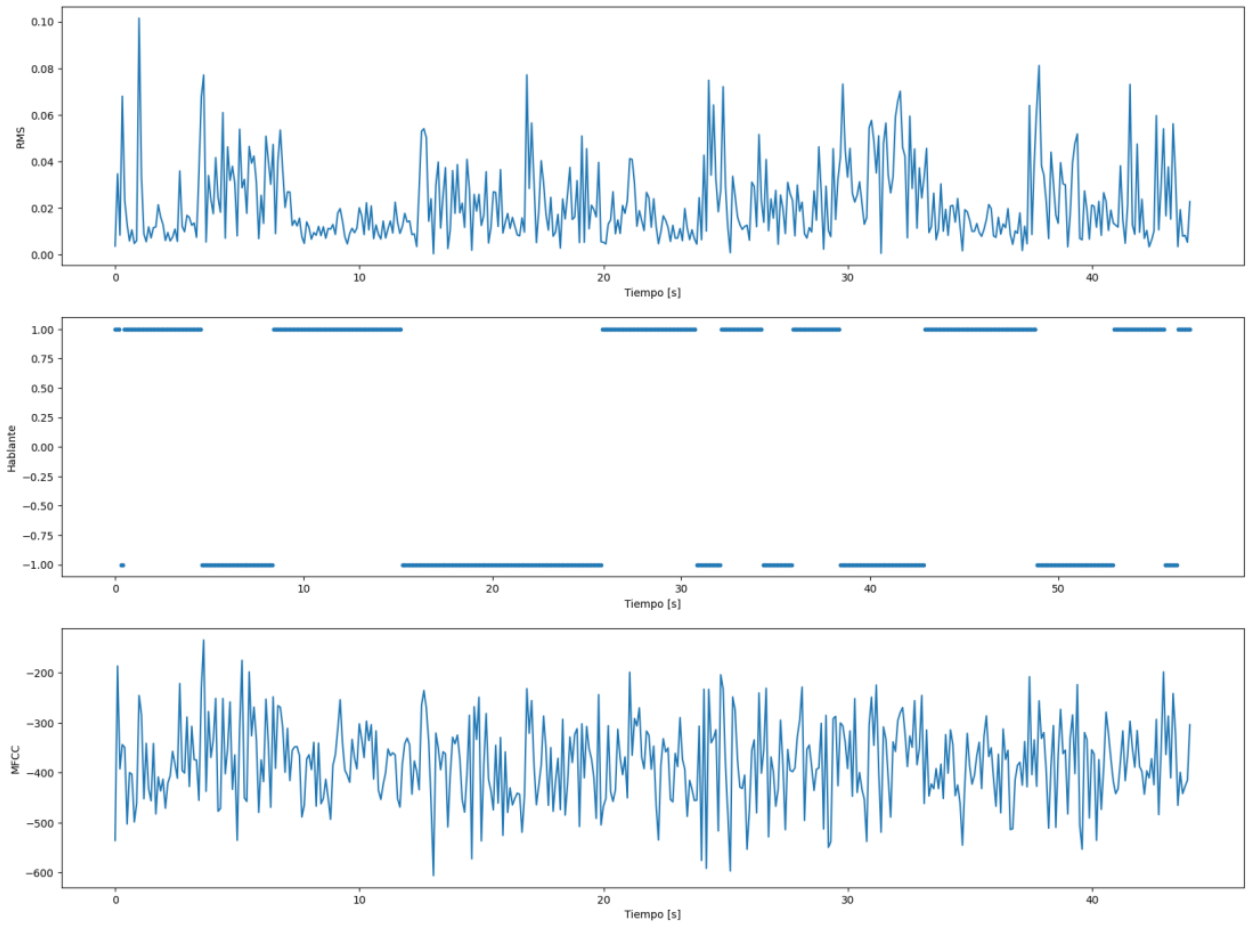

Figura 8: RMS, diarización y MFCC de la conversación

Lo realizado en el último punto se justifica debido a que la libreria pydub elimina todas las ondas bajo cierto umbral de decibelios (dB), y existe una parte al comienzo y al final de cada frase que es eliminada por ser confundida con el ruido de fondo.

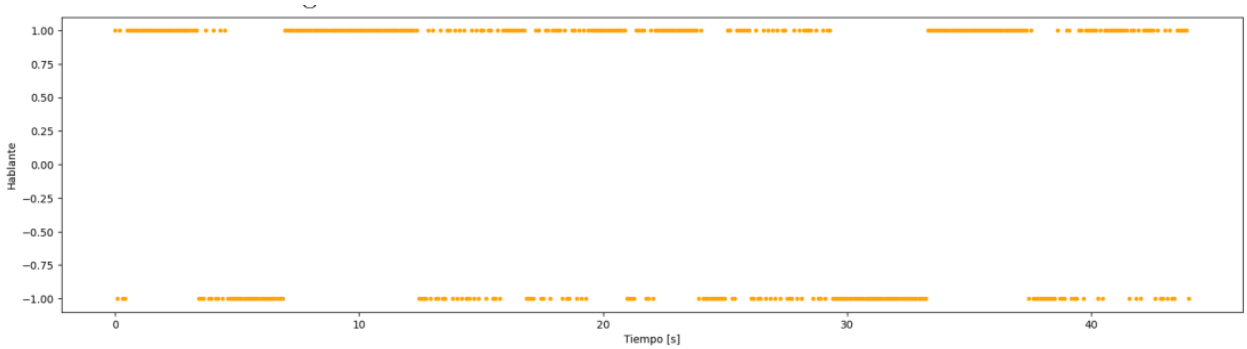

Figura 9: Diarización calculada con RMS suavizada

Luego de realizar los pasos descritos anteriormente tanto para RMS como para MFCC, se procedió a buscar la intersección de cada una de las diarizaciones, con el objetivo de calcular una diarización de mayor exactitud y veracidad.

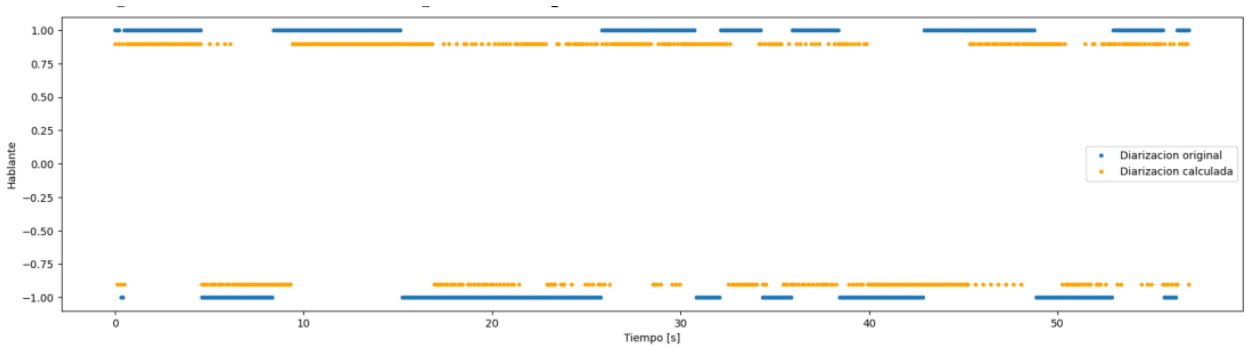

Figura 10: Diarización original comparada con la diarización calculada con RMS

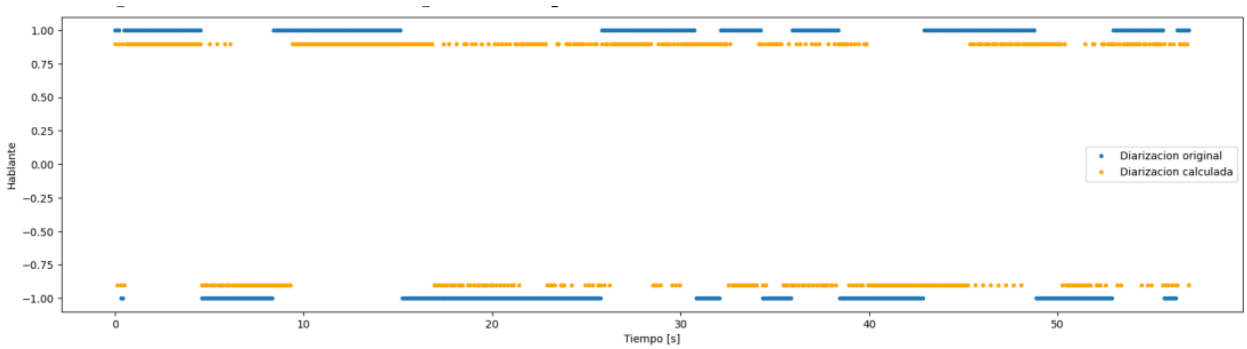

Figura 11: Diarización original comparada con la diarización intersección de RMS y MFCC

Los puntos en cero de este gráfico corresponden a las zonas de la curva donde no se intersectan las diarizaciones calculadas con RMS y MFCC, con lo cual, podemos inferir que el hablante está indeterminado en esos puntos.

4.2. Limpieza de Ruido

Para comenzar, tenemos que identificar los tipos de ruidos más comunes dentro de los audios de grabaciones de llamadas telefónicas. Luego de aislar dichos ruidos, se debió crear un dataset de entrenamiento y validación para plantear una solución con inteligencia artificial. Finalmente, fue necesario crear un modelo de red neuronal capaz de quitar el ruido de los archivos de audio ruidosos.

4.2.1. Identificar los ruidos

Lo primero fué conseguir un conjunto de audios ruidosos para ser analizados. Alloxentric es una empresa que tiene como uno de sus productos AnalíticaCX, el cuál está encargado de evaluar grabaciones de llamadas telefónicas de call centers, midiendo que los ejecutivos que realicen las llamadas sigan correctamente un guión. Esto significa que mientras peor evaluada está la grabación, menor cercanía hay entre lo que dice el guión y la transcripción del audio.

Esta diferencia puede deberse tanto como a un error por parte del ejecutivo como a una mala transcripción del audio, y por esto, las grabaciones escogidas para el análisis debian cumplir con:

1. Audio de duración mayor a 60 segundos: Se buscan audios con una duración hablada de mayor a un minuto (sin contar silencios) para asegurarse que haya una interacción significativa entre los participantes.
2. Baja evaluación: Suponiendo que el ejecutivo seguirá el guión lo mejor posible, una baja evaluación puede ser signo de una mala transcripción.
3. Baja confianza de STT (Speech-To-Text): La plataforma de STT usada es Deepgram, y al transcribir un audio entrega un valor de confianza que refleja cierta seguridad del algoritmo con el resultado generado, y una baja confianza está relacionada con que el STT no dió una respuesta suficientemente correcta.

Con estos puntos, se seleccionaron 100 audios de la base de datos de Alloxentric para realizar un análisis manual de los posibles origenes de ruido en los archivos. Es apropiado destacar que se realizó un análisis cualitativo de las grabaciones, y no se utilizó ninguna característica facilmente medible.

- Distorciones (32/100): Audios que en algunas secciones presentaban una señal similar a la estática en una televisión que no tiene un canal sintonizado.
- Eco (10/100): Probablemente debido a alguna demora en la transmición de las voces, muchos audios presentaban eco proveniente del dispositivo del cliente, escuchándose una repetición de la voz del ejecutivo.
- Ruido de ambiente (6/100): Algunas de las llamadas telefónicas se realizaban en algún horario en donde el cliente se encontraba en la calle, dentro del transporte público, etc., y quedaron grabados sonidos de vehículos, de terceros que no participan de la llamada, entre otros.

Identificando que los principales problemas dentro de las grabaciones telefónicas provenian de distorciones aparentemente aletorias y del eco de alguno de los intergantes, se procedió a crear un dataset de entrenamiento y validación para el posterior uso de un modelo WaveNET.

4.2.2. Dataset de entrenamiento

Para esto, se seleccionaron audios de "buena calidad"de la base de datos de Alloxentric, los cuales, siguiendo los patrones de selección anterior, debian cumplir con:

1. Audio de duración mayor a 60 segundos.
2. Evaluación mayor a 80 %.
3. Confianza de STT mayor a 99 %.

Con esto se obtuvo un conjunto de 988 audios con los cuales crear el dataset. Este grupo de audios correspondian a casi 22 horas de grabaciones telefónicas de bajo ruido, y para realizar el entrenamiento de la red neuronal fueron divididos aleatoriamente en dos subconjuntos de 800 audios para la etapa de entrenamiento y 188 para la etapa de validación. (El dataset de prueba es un conjunto diferente que será definido en el siguiente capítulo)

Para esta parte, se utilizó la librería **pydub** con el objetivo de modificar los audios limpios para crear audios ruidosos. Los algoritmos utilizados fueron los siguientes

Audios con eco Para crear los audios con eco, se agregó una repetición de las voces con una demora de 500ms reduciendo la intensidad de la voz.

```
import os
from pydub import AudioSegment

listaAudios = os.listdir('./audios/audios_limpios')
for audio in listaAudios:
    mp3 = AudioSegment.from_file(f"./audios/audios_limpios/{audio}",
 format='mp3')
    mp3_eco = mp3.fade_in(500).apply_gain(-5)
    mp3_eco.export(f"./audios/audios_eco/{audio}", format='mp3')
```

Audios con distorciones Las distorciones de las grabaciones son difíciles de reproducir, debido a que parecen ser aleatorias, o provenientes de la reducción de muestreo del archivo de audio para obtener audios que pesen menos y sean más fáciles de almacenar. Para esto se plantearon dos alternativas, agregar señales aleatorias o reducir la tasa de muestreo de una grabación. El código en python para ambas es el siguiente:

```
import os
from pydub import AudioSegment
import numpy as np

def add_white_noise(audio, noise_level):
    # Genera el ruido blanco con la misma duracion que el audio
    duration = len(audio)
    noise = np.random.uniform(-1, 1, duration) * noise_level

    # Convierte el ruido en un objeto AudioSegment
    noise_audio = AudioSegment(
        noise.tobytes(),
        frame_rate=audio.frame_rate,
        sample_width=audio.sample_width,
        channels=audio.channels
    )

    # Mezcla el ruido con el audio original
    audio_with_noise = audio.overlay(noise_audio)

  return audio_with_noise

listaAudios = os.listdir('./audios/audios_limpios')
for audio in listaAudios:
    mp3 = AudioSegment.from_file(f"./audios/audios_limpios/{audio}",
 format='mp3')

    # Se crea la version con ruido
    mp3_ruido = add_white_noise(mp3, 5)
    mp3_ruido.export(f"./audios/audios_ruido/{audio}", format='mp3')

    # Se crea la version con baja tasa de muestreo
    mp3_low_sample = mp3.set_frame_rate(4000)
    mp3_low_sample.export(f"./audios/audios_low_sample/{audio}", format
 ='mp3')
```

En donde se escogió un nivel de ruido de 5 para los audios de ruido aleatorio, y fué definida una tasa de muestreo de 4000Hz para las versiones de bajo frame rate. La tasa original de los audios es de 8000Hz.

4.2.3. Creación de modelo WaveNET

Teniendo los datos para realizar el entrenamiento, ahora es necesario diseñar el modelo de inteligencia artificial para limpiar el ruido de los audios. Para esto utilizaremos la arquitectura de WaveNET que nos permite una alta eficienca al entrenar con archivos de sonido sin procesar.

Utilizando como ejemplo el trabajo realizado por un grupo de la Universidad Española Pompeu Fabra, el cuál se encuentra en el paper *A Wavenet for Speech Denoising* [4], se implementó usando las librerías de **pytorch** y **keras** una versión simplificada de la red neuronal propuesta en dicho informe. Cabe señalar que solo se utilizó la arquitectura para el diseño de la red neuronal, y no se usó una red pre-entrenada, sino que se comenzó a entrenar desde cero.

El modelo consta de una primera capa convolucional que crear un vector de características para cada punto, seguido de N bloques residuales que contienen las capas **Dilated Causal Convolutional** (DCC) y finaliza con una capa convolucional que vuelve a reducir la dimensionalidad para que sea igual a la del input.

Cada bloque residual contiene una capa DCC con un retraso o dilatación específico, junto a una capa de activación ReLu. Siguiendo la propuesta del paper original de WaveNET [3], la dilatación de cada capa DCC va aumentando como potencias de 2 formando ciclos al momento de llegar a 512. Por ejemplo, un modelo con 12 bloques residuales tendrá 12 capas DCC con valores de dilatación [2, 4, 8, ..., 256, 512, 2, 4].

```
from keras.models import Model
from keras.layers import Input, Conv1D, Activation, Add

def residual_block(x, dilation_rate, filters, kernel_size):
    # Capa convolucional dilatada
    conv = Conv1D(filters, kernel_size, padding='causal', dilation_rate=
 dilation_rate)(x)
    conv = Activation('relu')(conv)

    # Capa convolucional 1x1 para ajustar el numero de canales
    conv = Conv1D(filters, 1, padding='same')(conv)

    # Conexion residual
    x = Add()([x, conv])

    return x

def build_wavenet(input_shape, num_layers, num_filters, kernel_size):
    inputs = Input(shape=input_shape)

    # Capa inicial
    x = Conv1D(num_filters, kernel_size=1, padding='causal')(inputs)

```

```
    # Capas residuales
    skip_connections = []
    for i in range(num_layers):
        dilation_rate = 2 ** i
        x = residual_block(x, dilation_rate, num_filters, kernel_size)
        skip_connections.append(x)

    # Combinacion de las conexiones residuales
    x = Add()(skip_connections)
    x = Activation('relu')(x)

    # Capa de salida
    outputs = Conv1D(1, kernel_size=1, activation='linear', padding='
same')(x)

    model = Model(inputs=inputs, outputs=outputs)

    return model
```

Los valores de los hiperparámetros escogidos para el modelo fueron los siguientes:

- **input shape = (160000, 1)**: Los archivos de audio representados como listas de valores fueron recortados en fragmentos de 20 segundos para el entrenamiento, obteniendo listas de 160000 datos para las grabaciones de 8000Hz.
- **num layers = 30**: Este número define la cantidad de bloques residuales que tendrá el modelo.
- **num filters = 32**: Este parámetro define el número de características que las capas convolucionales pueden extraer de cada dato.
- **kernel size = 2**: El tamaño del kernel que usarán las distintas capas del modelo.

Esta modelo es una red neuronal pequeña, ya que está compuesta por 63552 parámetros, lo cual permitió un entrenamiento en tiempos de ejecución más acotados. El modelo fue entrenado por 150 épocas utilizando **batch size = 32**, obteniendo la siguiente curva de pérdida:

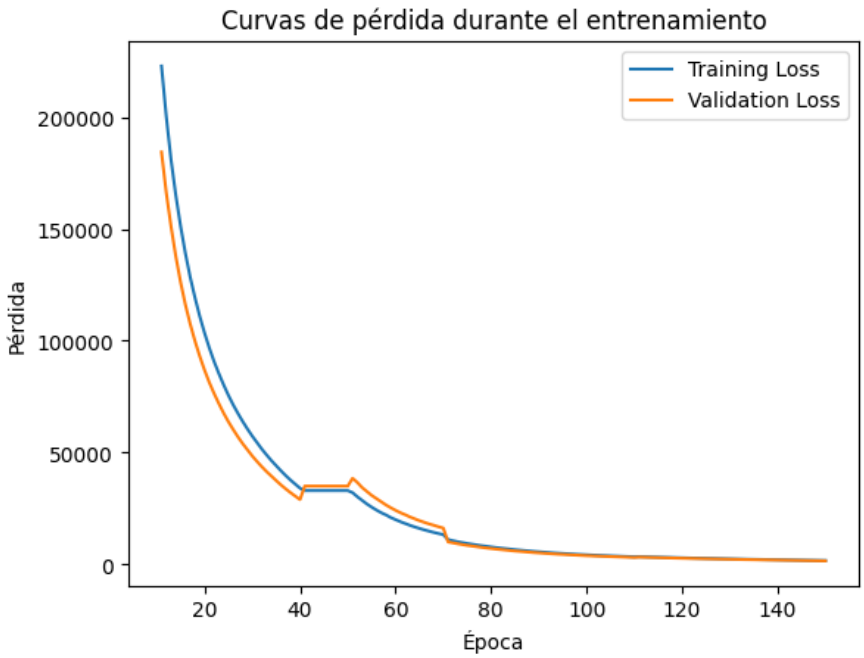

Figura 12: Curva de pérdida durante las 150 épocas del entrenamiento

Capítulo IV

5. Evaluación

Al igual que en la parte anterior, podemos dividir la evaluación de las soluciones propuestas en dos secciones diferentes.

5.1. Diarización

Para el dataset de 20 audios diarizados manualmente, se procedió a calcular la diarización considerando los siguientes puntos:

1. Diarización RMS: solo se utiliza el parámetro de RMS para calcularla.
2. Diarización MFCC: solo se utiliza el parámetro de MFCC para calcularla.
3. Diarización RMS x MFCC: se intersectan los dos resultados anteriores. Los puntos en donde no se intersecten estas curvas se consideran de indeterminación, ya que dos algoritmos dan respuestas diferentes.
4. Diarización Deepgram: la plataforma de Deepgram ofrece una diarización al momento de transcribir los audios, pero esa característica no está aun optimizada para grabaciones en español.

Los resultados promediados de los 20 audios analizados, considerando una confianza del 90 % y una distribución normal, fueron los siguientes:

- Acierto diarización RMS: 61.52 % (± 2.19 %)
- Acierto diarización MFCC: 63.11 % (± 3.84 %)
- Acierto diarización RMS x MFCC: 51.27 % (± 3.87 %)
- Diarización indeterminada RMS x MFCC: 22.61 % (± 2.28 %)
- Diarización Deepgram: 64.0 % (± 2.59 %)

Entre los cuales, los audios con mayor y peor evaluación fueron:

Audio mejor resultado

- Acierto diarización RMS: 72.3 %
- Acierto diarización MFCC: 79.1 %
- Acierto diarización RMS x MFCC: 70.1 %
- Diarización indeterminada RMS x MFCC: 12.5 %
- Diarización Deepgram: 85.0 %

Audio peor resultado

- Acierto diarización RMS: 53.3 %
- Acierto diarización MFCC: 45.3 %
- Acierto diarización RMS x MFCC: 33.2 %
- Diarización indeterminada RMS x MFCC: 32.0 %
- Diarización Deepgram: 52.5 %

De todos los parámetros que se analizaron, estos dos fueron los que mostraron una mayor variabilidad al cambiar el interlocutor dentro de la grabación de las llamadas telefónicas. Con esto, fue posible encontrar un valor limite que diferenciara a ambos integrantes de manera notable, pero aun está lejos de ser una herramienta que mejore las diarizaciones obtenidas por Deepgram.

La función de diarización de la plataforma Deepgram se encuentra bien optimizada para audios en inglés, lo cual indica que se pueden obtener mejores resultados con el entrenamiento de un modelo LSTM para la diarización de grabaciones telefónicas, pero esta fue una aproximación más simple que logró obtener unos resultados parecidos a los que entrega Deepgram actualmente en español, y aún hay espacio para mejora.

5.2. Limpieza de Ruido

Para realizar la evaluación de la red neuronal entrenada, se transcribieron manualmente 30 audios adicionales a los 20 usados en la sección anterior, formando un dataset de 50 audios. Este conjunto de audios es direrente al dataset de entrenamiento, y está compuesto por grabaciones con una duración entre 90 y 120 segundos y con una confianza de transcripción de Deepgram del 99 % (Lo que muy probablemente indica poco ruido). Para cada uno de estos audios se decidió comparar la calidad de las transcripciones realizadas por Deepgram en 3 estados:

- Audio Original: la versión de la grabación obtenida de la base de datos de Alloxentric.
- Audio Ruidoso: se aplica un filtro de ruido al archivo de audio para generar distorsiones.
- Audio Procesado: el audio ruidoso es corregido con la red neuronal WaveNET entrenada.

El filtro de ruido que se escogió para la evaluación es la aplicación de **eco** a las grabaciones, ya que es el modelo entrenado que cualitativamente generaba audios más limpios, en comparación a las otras dos redes neuronales entrenadas.

Para medir el nivel de cercanía de cada versión transcrita con la transcripción manual, se creó un algoritmo que permite identificar cuantas palabras exactas fueron transcritas correctamente, cuantas palabras se parecen a la palabra exacta que debia haber sido transcrita, cuantas palabras no fueron identificadas en la transcripción y el número de palabras nuevas que el STT generó pero no se encontraban en el audio original.

DEEPGRAM

Aló? Muy buenos días. Buenos días. Buenos días. ¿Me este número con XXXX XXXX XXXX XXXX? Con él. XXXX XXXX usted habla con XXXX XXXX. Yo le llamo de XXXX, gusto saludarlo. Don XXXX XXXX, lo lo contactamos en esta oportunidad, por cargo de XXXX XXXX. Únicamente para consultar, yo confirmar una cuenta con fecha cinco de JERIA, que correspondería a un plan de pago. ¿El usted había efectuado recientemente este pago bajo XXXX XXXX o había o normalmente realizó el pago de la misma semana? Jueves que se duele que que ir a un correo, me dijeron, pues no no había dicho nada. Un correo. Usted va a pagar directamente a la sucursal o por transferencia? Es que no sé, yo creo que la próxima semana vez me puede acercar a a la sucursal, ¿pero qué cuánto el valor? El monto que a mí me parece de 71877 pesos. No, yo la semana me acercó. ¿Tiene usted posibilidad de ir el día martes de la próxima semana, martes? No, yo tengo que mil con fue uno dos dos. Perfecto. Voy registrada la información XXXX XXXX. Ya, yo qué hago ya llegó allá nomás? Sí, pasa con un ejecutivo y informe le informe que quiere realizar el pago de un de va a realizar la cancelación de un plan de pago, se va cuota del plan de pago. Ya, no hay problema. Ajá. Pero tiene que pasar con el ejecutivo primero? Sí, sí, sí hay problema. Muy bien, con XXXX, le recover señorita, muchas gracias haberlo ofrecido. Te lo digo.

TRANSCRIPCION MANUAL

Alo, alo, muy buenos dias, buenos dias, buenos dias, ubico en este numero a don XXXX XXXX XXXX XXXX, con el, don XXXX XXXX usted habla con XXXX XXXX y yo lo llamo de XXXX gusto en saludarlo, don XXXX XXXX lo contactamos en esta oportunidad por encargo de XXXX XXXX unicamente para consultar cierto confirmar una cuenta con fecha cinco de julio que corresponderia a un plan de pago, el, usted habia efectuado recientemente este pago don XXXX XXXX o habia o normalmente realiza el pago dentro de la misma semana del, no porque se supone que tenia que llegar un correo me dijieron, no he dicho nada, un correo, solo un correo, usted va a pagar directamente a la sucursal o por transferencia, es que no se po, yo creo que la proxima semana me podria acercar a la sucursal pero que cuanto es el valor, el monto que a mi me aparece es de 71787 pesos, si no, yo la otra semana me acerco, tiene usted la posibilidad de ir el dia martes de la proxima semana martes, no yo creo que entre el miercoles y el jueves uno de los dos, perfecto, voy a dejar recordada la informacion XXXX XXXX, ya yo que hago ya llego alla nomas, si pasa por un ejecutivo y informa le informa que desea realizar el pago de el pago de un banco, va a realizar la cancelacion de un plan de pago, o va a pagar, ya, cuotas del plan de pago, ya no hay problema, bien pero tiene que pasar con el ejecutivo primero, si si no hay problema, muy bien don XXXX XXXX dejo el registro entonces, muchas gracias, hasta luego, gracias por habernos atendido, hasta luego.

Figura 13: Ejemplo con una transcripción de como se identifica la presición del STT

En este caso, de la imagen 13 podemos notar que las palabras con fondo verde son las Palabras exactas, las Palabras similares estan en color verde sin fondo, las Palabras no transcritas serian las de color rojo en el párrafo manual y las Palabras creadas por el STT son las de color rojo en el párrafo de Deepgram.

- Palabras exactas: 213 de 289 (73.7 %) se calculan en base al total de palabras en el párrafo de transcripción manual.
- Palabras similares: 7 de 289 (2.4 %) se calculan en base al total de palabras transcritas manualmente y junto a las palabras exactas se consideran parte de una transcripción exitosa.
- Palabras no transcritas: 67 de 289 (23.2 %) son las palabras de la transcripción manual que no se encontraron en el párrafo generado por el STT.
- Palabras creadas por STT: 40 de 260 (15.4 %) son las palabras generadas por el STT que no se encuentran en la transcripción manual.

En este caso, la definición de palabra similar se realiza con la librería de python **Levenshtein** de la siguiente manera:

```
1  import Levenshtein as lv
2
3  def similares(palabra_1, palabra_2)
4    similaridad = lv.ratio(palabra_1, palabra_2)
```

```
    if similaridad > 0.9:
      # Las palabras son similares
      return True
    else:
      # Las palabras son muy diferentes
      return False
```

Vale la pena señalar que para considerar que el procedimiento establece una mejora en la calidad de la transcripción es necesario que el porcentaje de Palabras exactas o Palabras similares aumente, mientras que el porcentaje de Palabras no transcritas o creadas por el STT deberían disminuir.

El promedio de los resultados obtenidos por los 50 audios analizados de esta forma se encuentra en la tabla 1, los cuales correspondian a grabaciones de buena calidad a las que se les aplicó un filtro de ruido (eco).

Versión Audio	Palabras exactas	Palabras similares	Palabras no transcritas	Palabras creadas por STT
Original	64.72 ± 1.43 %	5.97 ± 0.27 %	27.57 ± 0.46 %	22.40 ± 0.51 %
Ruidoso (ECO)	58.33 ± 2.02 %	8.07 ± 0.26 %	29.86 ± 0.57 %	25.45 ± 0.62 %
Procesado	60.95 ± 1.73 %	1.49 ± 0.21 %	36.95 ± 0.67 %	20.47 ± 1.08 %

Tabla 1: Tabla de audios originales sin ruido que fueron procesados

Al igual que en la sección anterior, los intervalos de confianza se calcularon considerando una confianza del 90 % y una distribución normal.

En los datos anteriores podemos notar como el ruido disminuyó el número de palabras exactas encontradas, pero los resultados obtenidos luego de procesar los audios muestran que la tasa de éxito de la transcripción termina siendo inferior si consideramos la suma de las palabras exactas y las similares. Realizando un análisis cualitativo de algunos de los audios, se puede ver una disminución en la intensidad de la voz de los hablantes, lo cual puede generar que el STT de Deepgram no logre encontrar esas palabras. Al oido humano, los audios se escuchan más "limpios", pero parece que el estado actual del modelo no es suficiente para ver una mejora en las transcripciones.

Además de los audios anteriores, que estaban considerados limpios y se les aplicó un filtro de ruido, se escogieron 20 audios del conjunto de 100 grabaciones ruidosas (ver 4.2.1), se transcribieron manualmente y tambien fueron procesados por la red neuronal para comprobar si existía una mejora, y los resultados se encuentran en la tabla 2.

Versión Audio	Palabras exactas	Palabras similares	Palabras no transcritas	Palabras creadas por STT
Ruidoso	58.09 ± 1.42 %	3.27 ± 0.67 %	38.06 ± 1.75 %	36.46 ± 1.1 %
Procesado	60.6 ± 2.36 %	9.10 ± 1.44 %	28.52 ± 2.24 %	31.57 ± 2.12 %

Tabla 2: Tabla de audios ruidosos que fueron procesados

En el análisis de los audios que eran originalmente ruidosos, podemos ver unos resultados más prometedores. Si bien el número de palabras exactas no varía mucho, las palabras

similares aumentaron considerablemente

Capítulo V

6. Conclusiones

Este trabajo de investigación ha presentado como los últimos avances en las tecnologías de software pueden ayudarnos a resolver situaciones específicas muy complejas que no podrían ser logradas aplicando algoritmos más tradicionales. El desarrollo de las redes neuronales de tipo WaveNET, nos abren posibilidades increibles en el análisis de audios, sobre todo en un área como la telefonía, donde la calidad de las señales no es siempre la mejor.

El trabajo total de esta investigación fueron dos semestres. Durante el primero se sentaron las bases del proyecto, se recolectó información sobre el estado del arte en el mundo del análisis de audio, y se creó un algoritmo que permite identificar a los participantes dentro de una conversación. Aunque sus resultados no fueron los esperados, ya que no superó a los métodos actuales en diarización, si establece una base de lo que se puede lograr adoptando técnicas más avanzadas, y muestra que características son las que más ayudan a la identificación de los integrantes de una conversación. Con esto, podemos concluir que el primero de los objetivos específicos fue cumplido parcialemnte, ya que si bien se creó un algoritmo de diarización, este es muy simple, y sus resultados están por debajo de lo que logra la aplicación de Deepgram actualmente.

Durante el segundo semestre de trabajo, el enfoque fue principalmente a los siguientes objetivos específicos, que correspondian a identificar las fuentes de ruido más comunes y entrenar un modelo de red neuronal que mejore su calidad. El aislamiento del ruido en un audio es una tarea muy difícil, ya que mucho se basa en percepciones cualitativas de lo que es un ruido dentro de una pieza de sonido. En esta investigación se definieron dos tipos de ruido principales, eco y distorsión, y si bien fue posible generar una recreación satisfactoria de dichas alteraciones, es un área que está abierta a nuevas definiciones e interpretaciones de que elementos corresponden a ruidos y como poder simularlos.

Para el diseño e implementación del modelo de inteligencia artificial, después de entender el estado actual de la industria tecnológica en materia de análisis de audios, se escogió el modelo WaveNET para realizar este trabajo, y con él fue posible cumplir con el objetivo específico planteado, pero aún queda trabajo por hacer para poder alcanzar el objetivo general que era mejorar las transcripciones.

Con la definición de un modelo sencillo de red neuronal fue posible obtener importantes resultados en como el mejoramiento de una pieza de audio puede lograr una transcripción más precisa, por lo que con más tiempo y mejores recursos de hardware será posible seguir con este trabajo como base para implementar redes neuronales más grandes con datasets de mayor calidad y así lograr una mejora significativa dentro del análisis de grabaciones telefónicas.

Los resultados actuales quedaron un poco por debajo de las expectativas planteadas al principio del proyecto, ya que las mejoras en audios ruidosos es poco significativa (los intervalos de confianza en los resultados se solapan). Aún asi, observando los resultados de los audios ensuciados artificialmente (se les agrega una capa de ruido), estos muestran que la

alternativa escogida tiene mucho potencial. Con algunos meses de trabajo y con un dataset pequeño de menos de 22 horas de grabaciones telefónicas, fue posible aumentar tanto el porcentaje de palabaras exactas transcritas, como el porcentaje de palabras similares, y esto nos enseña del impacto que este algortimo puede tener dentro de la industria de transcripciones y análisis de audios.

Trabajos futuros que podrian realizarse a partir de lo desarrollado en este investigación son la implementación de una red neuronal WaveNET con más capas y parámetros, ya que las limitantes de hardware fueron importantes para las elecciones que se tomaron en este trabajo. Se podría pasar de un modelo pequeño de 60000 parámetros a otro con millones de ellos, permitiéndonos representar un conjunto más grande de características para las señales de audio. Además de esto, otra mejora significativa sería incrementar el tamaño del dataset de entrenamiento. Con las limitaciones de una persona, se transcribieron manualmente menos de 100 audios en el transcurso de estos 2 semestres, y la recolección de audios ruidosos es una tarea lenta que no permitió tener una base más grande. De esta forma, resolviendo los problemas de escala al definir los datos de entrenamiento, este algoritmo tiene un potencial muy grande para mantenerse relevante y mejorar la calidad de la industria del análisis de audio.

BIBLIOGRAFÍA

[1] Bredin, Hervé, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz y Marie Philippe Gill: *pyannote.audio: neural building blocks for speaker diarization*. En *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[2] Deepgram. 2022. `https://developers.deepgram.com/api-reference/`.

[3] Oord, Aaron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior y Koray Kavukcuoglu: *WaveNet: A Generative Model for Raw Audio*, 2016.

[4] Rethage, Dario, Jordi Pons y Xavier Serra: *A Wavenet for Speech Denoising*, 2018.

[5] Sainburg, Tim: *timsainb/noisereduce: v1.0*, Junio 2019. `https://doi.org/10.5281/zenodo.3243139`.

[6] Sainburg, Tim, Marvin Thielk y Timothy Q Gentner: *Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires*. PLoS computational biology, 16(10):e1008228, 2020.

[7] Schröter, Hendrik, Alberto N. Escalante-B., Tobias Rosenkranz y Andreas Maier: *DeepFilterNet: A Low Complexity Speech Enhancement Framework for Full-Band Audio based on Deep Filtering*. En *ICASSP 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[8] Wang, Quan, Carlton Downey, Li Wan, Philip Andrew Mansfield y Ignacio Lopez Moreno: *Speaker Diarization with LSTM*, 2022.